INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Factoring algorithm for counting the number of (s,t)-mincuts of each size

Stéphane Bulteau

N° 3209

Juillet 1997

THÈME 1

Rapport de recherche

ISSN 0249-6399

INRIA
RENNES

Factoring algorithm for counting the number of (s,t)-mincuts of each size

Stéphane Bulteau*

Thème 1 — Réseaux et systèmes

Projet Model

Rapport de recherche n ° 3209 — Juillet 1997 — 19 pages

Abstract: An efficient family of methods to evaluate network reliability is the class of factoring algorithms. Their efficiency is due to the use of reliability-preserving reductions (for instance, series-parallel ones). In this work, we follow a similar approach for the problem of counting the (s,t)-mincuts of a graph with size i, for all i. In order to take into account these possible reductions, the paper presents a technique based on a factoring formula, similar to the relation of Moskowitz, to count the number of (s,t)-mincuts of each size. This formula is applied recursively and various reductions, with computational cost polynomial in the size of the network, are used in order to reduce the size of the studied graph.

Key-words: (s,t)-minimal cutset, factoring algorithm, reliability-preserving reductions

(Résumé : tsvp)

* Email: sbulteau@irisa.fr

Un algorithme de factorisation pour compter le nombre de (s,t)-coupes minimales de chaque taille

Résumé : Une famille de méthodes efficaces pour évaluer la fiabilité des réseaux est la classe des algorithmes de factorisation. Leur efficacité est due à l'utilisation possible de réductions préservant la fiabilité (par exemple les réductions séries-parallèles). Dans ce travail, nous suivons une approche similaire pour le problème du comptage des (s,t)-coupes minimales de taille i, pour tout i, dans un graphe donné. Afin de pouvoir tenir compte de ces réductions possibles, ce papier présente une technique basée sur une formule de factorisation, similaire à la relation de Moskowitz, pour compter le nombre de (s,t)-coupes minimales de chaque taille. Cette formule est appliquée récursivement et diverses réductions, de complexité polynomiale dans la taille du réseau, sont utilisées pour réduire la taille du réseau étudié.

Mots-clé : (s,t)-coupe minimale, algorithme de factorisation, réductions préservant la fiabilité

1 Introduction

This paper considers the problem of counting all the (s,t)-mincuts of each possible size separating two specified vertices s and t in an undirected graph. In [1], in order to measure the (s,t)-vulnerability of a network, we have proposed the vector of (s,t)-mincuts whose ith component is the number of (s,t)-mincuts of size i. It has been shown that this measure verifies several interesting properties, similar to some properties of the reliability measure. The combination of these properties makes possible to use series-parallel reductions and decomposition into biconnected and triconnected components to evaluate the vulnerability of a network.

In order to generate (and then count) all the (s,t)-mincuts in a graph, one of the most efficient known method is the paradigm of Provan and Shier [2], similar to the method of Tsukiyama and *al.* presented in [3]. It is based on a pivotal decomposition on a vertex. However, for a class of undirected planar graphs, called delta-star reducible graphs, Sung and Yoo proposed a new approach [4], more efficient, based on series-parallel reductions to enumerate all the (s,t)-mincuts. In [5], Ramanathan and Colbourn used an edge-factoring algorithm, without reductions, to count almost all the mincuts. In a different framework, one of the most efficient method in exact reliability evaluation is based on an edge-factoring algorithm with various reliability-preserving reductions, such as series-parallel reductions [6]. These results lead us to propose an edge-factoring algorithm to count all the (s,t)-mincuts of each size. Reductions with polynomial cost are applied on each network generated during the recursive factorization process, in order to reduce its size. We show that the paradigm of Provan and Shier can be considered as an edge-factoring algorithm without series-parallel reductions.

Section 2 gives some definitions and notation used in the paper. It also presents some properties of the vector of (s,t)-mincuts. In Section 3, we describe some mincut-preserving reductions which can be employed with the factoring algorithm to reduce the computational time. Section 4 presents the factoring formula and the induced algorithm. We compare, in Section 5, our algorithm with the paradigm of Provan and Shier [2]. Section 6 is devoted to some conclusions.

2 Preliminaries

We resume in this section some notation and definitions.

- Let $\mathcal{G} \equiv (\mathcal{V}, \mathcal{E})$ be an undirected graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$.
- We denote by e_i or by e an edge of $\mathcal{E}$ and by (u, v) an edge between the two vertices u and v.
- $\mathcal{G} \bullet e$ is the graph deduced from $\mathcal{G}$ by deleting the edge $e \equiv (u, v)$ and contracting the vertices u and v into one single vertex.
- $\mathcal{G} - e$ is the graph obtained by deleting the edge e.
- Let $X \subset \mathcal{V}$ and $\mathcal{E}_X \equiv \{(u, v) \in \mathcal{E}$ such that $u, v \in X\}$. We denote by $\langle X \rangle \equiv (X, \mathcal{E}_X)$ the subgraph of $\mathcal{G}$ generated from X.

An (s, t)-*cut* is a set of edges, denoted by $(X, \overline{X})$, where X is a subset of vertices containing either s or t but not both and $\overline{X}$ is its complement; it is defined as the subset of edges having one extremity in X and the other one in $\overline{X}$. When the edges $(X, \overline{X})$ are removed, there is no path from s to t in the graph. A *cut* is an (s, t)-cut for some pair of nodes s and t. An (s, t)-*mincut* is an (s, t)-cut which does not contain any other (s, t)-cut. Let $(X, \overline{X})$ be an (s, t)-cut; it is an (s, t)-mincut if and only if the two subgraphs $\langle X \rangle$ and $\langle \overline{X} \rangle$ are connected. More generally, for the sets of vertices S and T of $\mathcal{V}$, with $S \cap T \equiv \emptyset$, we define an (S, T)-*cut* as a cut, $(X, \overline{X})$, with $S \subseteq X$ and $T \subseteq \overline{X}$ (or $S \subseteq \overline{X}$ and $T \subseteq X$). An (S, T)-*mincut* is an (S, T)-cut which is minimal for the inclusion. An X-*tree* is any minimal subgraph of $\mathcal{G}$ that connects the vertices in X.

Definition 2.1 *The counting vector of (s, t)-mincuts, denoted by $V_{s,t}(\mathcal{G}) \equiv (n_0, n_1, n_2, \dots)$, is the vector where n_i represents the number of (s, t)-mincuts having exactly i edges. We set $n_0 \equiv 0$ if $\mathcal{G}$ is (s, t)-connected and $V_{s,t}(\mathcal{G}) \equiv (1, 0, 0, \dots)$ otherwise (that is the empty set is the only (s, t)-mincut).*

This vector is used in [1] as a *vulnerability* measure.

In the same way, we can define the vector of (S,T)-mincuts of a network $\mathcal{G}$ by the vector $(n_0, n_1, n_2, \dots)$ where n_i denotes the number of (S,T)-mincuts having i edges. Here, we have to consider two particular cases. The first one is, as previously, when the empty set is an (S,T)-mincut, that is $\mathcal{G}$ is disconnected with S included in one connected component and T included in an other one. Then, we define $V_{S,T}(\mathcal{G}) \equiv (1,0,0,\dots)$. The second one is when there is no (S,T)-mincut, that is: $S \cap T \neq \emptyset$, or there is no S-tree, or there is no T-tree, or every S-tree and T-tree have at least one common point. In this case, we define $V_{S,T}(\mathcal{G}) \equiv (0,0,0,\dots)$.

Let us introduce the two operators $+$, the sum, and $*$, the convolution between two vectors.

If $U \equiv (U_0, U_1, \dots)$ and $V \equiv (V_0, V_1, \dots)$, then we have

$W^s \equiv U + V \equiv (W_0^s, W_1^s, \dots)$ with $W_j^s \equiv U_j + V_j$ and

$W^c \equiv U * V \equiv (W_0^c, W_1^c, \dots)$ with $W_n^c \equiv \sum_{j+k=n} U_j V_k$.

Let us switch to weighted graphs. Let $(\mathcal{G}, w)$ be a weighted graph where, for all edge e, $w(e)$ is a vector of integers. Let us define $w(\Gamma) = \mathop{\ast}_{e\in\Gamma} w(e)$ and

$$V_{S,T}(\mathcal{G}, w) \equiv \sum_{\text{all } (S,T)\text{-mincut } \Gamma} w(\Gamma).$$

Then, we have the following result:

Theorem 2.2 *[1] Given a graph $\mathcal{G}$ with, for each edge e, $w(e) \equiv (0,1,0,\dots)$, we have*

$$V_{S,T}(\mathcal{G}) = V_{S,T}(\mathcal{G}, w).$$

This theorem has an important consequence. It makes possible to evaluate the vector of (S,T)-mincuts in a graph $\mathcal{G}$ by evaluating the quantity $V_{S,T}(\mathcal{G}, w)$, with $w(e) \equiv (0,1,0,\dots)$ for each edge e. In this weighted graph, we can easily modify the topology without changing $V_{S,T}(\mathcal{G}, w)$, simply by changing the weights on edges. For example, we can replace two parallel edges e_i, e_j, by one edge e_k weighted by $(0,0,1,0,\dots)$: if e_i is in a mincut $\Gamma = (X, \overline{X})$ (which implies $e_j \in \Gamma$), then we have $e_k \in \Gamma' = (X, \overline{X})$. We obtain that $w(\Gamma) = w(\Gamma')$ and $V_{S,T}(\mathcal{G}, w) = V_{S,T}(\mathcal{G}', w')$ (for more details about this type of simplifications, the reader can see [1]). We will discuss in Section 3 on some reductions that make possible

to evaluate $V_{S,T}(\mathcal{G}, w)$ on an associated graph $(\mathcal{G}', w')$, by evaluating $V_{S,T}(\mathcal{G}', w')$. In the following sections, we will always work on this weighted graph $(\mathcal{G}, w)$.

3 Mincut-preserving reductions

The problem of evaluating the number of (S,T)-mincuts is exponential in the size of the network [7]. So, it is important to be able to reduce the size of the graph. To this aim, we identify here a family of mincut-preserving reductions, denoted by *MP*. These reductions are polynomial in the size of the network. They allow to count the (S,T)-mincuts in a graph $\mathcal{G}$ from the (S',T')-mincuts in a reduced network. These *MP* reductions are similar to the reliability-preserving reductions presented in [6]. In the latter case, they permit to solve the problem of the reliability evaluation in polynomial time [8], in series-parallel graphs. These reductions will be employed in the factoring algorithm presented in Section 4, in order to reduce the size of the involved graphs. The reductions transform the graph $\mathcal{G}$ into a new graph $\mathcal{G}'$ with a different topology and new values for the edges $e \in \mathcal{E}'$. We then obtain that the vector of mincuts $V_{S,T}(\mathcal{G})$ can be easily obtained from the vector of mincuts of $\mathcal{G}'$. That is, there exist a vector α and two sets of nodes S' and T' such that $V_{S,T}(\mathcal{G}) \equiv \alpha * V_{S',T'}(\mathcal{G}')$. We present here seven reductions and the decomposition into triconnected components which can also be considered as a reduction. For some of them, we have to contract the two extremities of an edge $e = (u,v)$ into a node w. If at least one of these nodes is in S (resp. T), then this set of vertices has to be modified. Then, we define for any set of vertices X, $X_e = X \cup \{w\} - \{u,v\}$ if $u,v \in X$ and $X_e = X \cup \{w\} - \{u\}$ if $u \in X$. We can remark that, if $u \in S$ and $v \in T$, then $w \in S_e$ and $w \in T_e$, which implies that there is no S_e, T_e-mincut. Based on Theorem 2.2 and the fact that we can easily deduce the S,T-mincuts in $\mathcal{G}$ from the S,T-mincuts in $\mathcal{G}'$, the proofs of the mincut-preserving reductions presented here are very simple. So, we will just give the proof of the first one.

MP_1: *Parallel reduction*

Let $e_i \equiv (u,v)$ and $e_j \equiv (u,v)$ be two edges in parallel in $\mathcal{G}$. Then, $\mathcal{G}'$ is obtained by replacing the two edges with a single edge e_k valued by $w(e_k) \equiv w(e_i) * w(e_j)$. We have $V_{S,T}(\mathcal{G}) = V_{S,T}(\mathcal{G}')$.

Proof. It is trivial that there is a one-to-one correspondence between the S,T-mincuts in $\mathcal{G}$ and the S,T-mincuts in $\mathcal{G}'$, given by:

$$\Gamma \equiv (X, \overline{X}) \text{ is an } S,T\text{-mincut in } \mathcal{G} \Longleftrightarrow \Gamma' \equiv (X, \overline{X}) \text{ is an } S,T\text{-mincut in } \mathcal{G}'.$$

If $e_i \in \Gamma$ (and $e_j \in \Gamma$), then $e_k \in \Gamma'$. In this case, we have

$$w(\Gamma) = w(e_i) * w(e_j) * w(\Gamma - \{e_i, e_j\}) = w(e_k) * w(\Gamma' - \{e_k\}) = w(\Gamma'),$$

as $\Gamma - \{e_i, e_j\} = \Gamma' - \{e_k\}$.

If $e_i \notin \Gamma$, then we have $w(\Gamma) = w(\Gamma')$, as $\Gamma = \Gamma'$. Then, we have $V_{S,T}(\mathcal{G}) = V_{S,T}(\mathcal{G}')$. ■

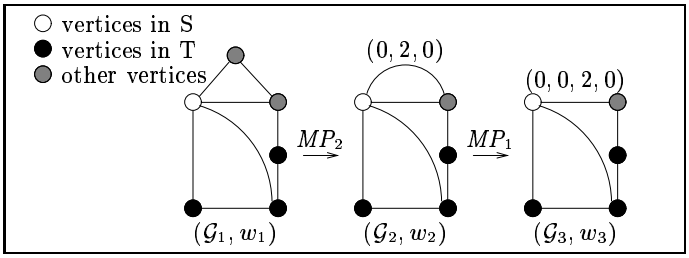

Figure 1: Example of series-parallel reductions with $V_{S,T}(\mathcal{G}_1) = V_{S,T}(\mathcal{G}_2) = V_{S,T}(\mathcal{G}_3)$

MP_2: Series reduction

Let $e_i \equiv (u,v)$ and $e_j \equiv (v,w)$ be two edges in series in $\mathcal{G}$, with $v \notin S \cup T$ and $deg(v) \equiv 2$. Then, $\mathcal{G}'$ is obtained by replacing the two edges with a single edge $e_k \equiv (u,w)$ valued by $w(e_k) \equiv w(e_i) + w(e_j)$ and deleting the node v. We have $V_{S,T}(\mathcal{G}) = V_{S,T}(\mathcal{G}')$.

MP_3: Bridge contraction

Let $e \equiv (u,v)$ be a bridge that occurs in all paths from a node in S to a node in T. Then, $\mathcal{G}' = \mathcal{G} \bullet e$ and we have $V_{S,T}(\mathcal{G}) = w(e) + V_{S,T}(\mathcal{G}')$. If $u \in S$, we obtain $V_{S,T}(\mathcal{G}) = w(e) + V_{S_e,T}(\mathcal{G}')$ and, if $u \in S$ and $v \in T$, we obtain $V_{S,T}(\mathcal{G}) = w(e)$, as $V_{S_e,T_e}(\mathcal{G}') \equiv (0,0,0,\dots)$.

MP_4: Trivial reduction

Let $e \equiv (u,v)$ with $u \in S$ and $v \in T$. Then, $\mathcal{G}' = \mathcal{G} - e$ and $V_{S,T}(\mathcal{G}) = w(e) * V_{S,T}(\mathcal{G}')$.

MP_5: *Vertex contraction*

Let $e \equiv (u, v)$ with $u, v \in S$ (or $u, v \in T$). We can contract u and v into a node w. Let $S' = S \cup \{w\} - \{u, v\}$. Then, $\mathcal{G}' = \mathcal{G} \bullet e$ and $V_{S,T}(\mathcal{G}) = V_{S',T}(\mathcal{G}')$.

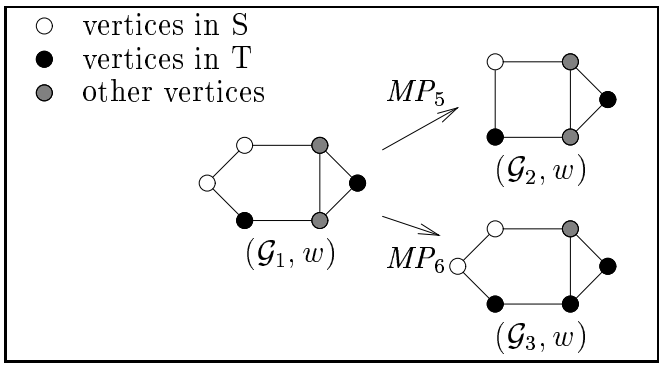

Figure 2: Example of vertex reductions with $V_{S,T}(\mathcal{G}_1) = V_{S,T}(\mathcal{G}_2) = V_{S,T}(\mathcal{G}_3)$

MP_6: *Vertex addition*

If $\langle \overline{S} \rangle$ has a cutvertex v for which at least two biconnected components contain a vertex in T, this cutvertex v can be added in T. We have $\mathcal{G} = \mathcal{G}'$ and $V_{S,T}(\mathcal{G}) = V_{S,T\cup\{v\}}(\mathcal{G}')$.

MP_7: *Irrelevant component deletion*

Let v a cutvertex of $\mathcal{G}$ and $\mathcal{G}^0 \equiv (\mathcal{V}^0 \cup \{v\}, \mathcal{E}^0)$, $\mathcal{G}^1 \equiv (\mathcal{V}^1 \cup \{v\}, \mathcal{E}^1)$ the two complementary connected components of $\mathcal{G}$ having v as a common vertex. There are nine possibilities:

1. $S \subseteq \mathcal{V}^0$ and $T \subseteq \mathcal{V}^0$ (or $S \subseteq \mathcal{V}^1$ and $T \subseteq \mathcal{V}^1$)
2. $S \subseteq \mathcal{V}^0$ and $T \subseteq \mathcal{V}^1$ (or $S \subseteq \mathcal{V}^1$ and $T \subseteq \mathcal{V}^0$)
3. $S \subseteq \mathcal{V}^0$, $T \cap \mathcal{V}^0 \neq \emptyset$ and $T \cap \mathcal{V}^1 \neq \emptyset$ (or $S \subseteq \mathcal{V}^1$, $T \cap \mathcal{V}^0 \neq \emptyset$ and $T \cap \mathcal{V}^1 \neq \emptyset$; $T \subseteq \mathcal{V}^0$, $S \cap \mathcal{V}^0 \neq \emptyset$ and $S \cap \mathcal{V}^1 \neq \emptyset$; $T \subseteq \mathcal{V}^1$, $S \cap \mathcal{V}^0 \neq \emptyset$ and $S \cap \mathcal{V}^1 \neq \emptyset$)
4. $S \cap \mathcal{V}^0 \neq \emptyset$, $S \cap \mathcal{V}^1 \neq \emptyset$, $T \cap \mathcal{V}^0 \neq \emptyset$ and $T \cap \mathcal{V}^1 \neq \emptyset$

In the first case, the component $\mathcal{G}^1$ is irrelevant and a reduction is obtained by deleting it; that is $\mathcal{G}' = \langle \mathcal{V}^0 \rangle$. Any self loop is irrelevant and its deletion is included in this category. We have $V_{S,T}(\mathcal{G}) = V_{S,T}(\mathcal{G}')$.

In the second case, the graph $\mathcal{G}$ can be split into two subgraphs $\mathcal{G}_0$ and $\mathcal{G}_1$. Then, we have $V_{S,T}(\mathcal{G}) = V_{S,v}(\mathcal{G}_0) + V_{v,T}(\mathcal{G}_1)$

In the third case, a reduction is performed by deleting the component $\mathcal{G}^1$. The set T becomes $T' = (T - (T \cap \mathcal{V}^1)) \cup \{v\}$. We have $V_{S,T}(\mathcal{G}) = V_{S,T'}(\mathcal{G}')$.
In the fourth case, there is no (S,T)-mincuts and then, $V_{S,T}(\mathcal{G}) = (0,0,0,\dots)$

Triconnected decomposition can also be used to count the (S,T)-mincuts. This comes from the following theorem:

Theorem 3.1 *Let $\mathcal{G}$ a graph and $\{u,v\}$ a separating pair such that S and T belong to the same triconnected component with respect to $\{u,v\}$, which is denoted by $\mathcal{H} \equiv (\mathcal{V}_\mathcal{H}, \mathcal{E}_\mathcal{H})$. We denote by $\mathcal{K} \equiv (\mathcal{V}_\mathcal{K}, \mathcal{E}_\mathcal{K})$ the complement of $\mathcal{H}$ in $\mathcal{G}$. We have $\mathcal{V}_\mathcal{H} \cup \mathcal{V}_\mathcal{K} \equiv \mathcal{V}$, $\mathcal{V}_\mathcal{H} \cap \mathcal{V}_\mathcal{K} \equiv \{u,v\}$, $\mathcal{E}_\mathcal{H} \cup \mathcal{E}_\mathcal{K} \equiv \mathcal{E}$ and $\mathcal{E}_\mathcal{H} \cap \mathcal{E}_\mathcal{K} \equiv \emptyset$. Define the reduced graph $\mathcal{G}^r \equiv (\mathcal{V}_{\mathcal{G}^r}, \mathcal{E}_{\mathcal{G}^r})$ by replacing $\mathcal{H}$ in $\mathcal{G}$ by a single edge between u and v, with value $w((u,v)) \equiv V_{u,v}(\mathcal{H})$ (we have $\mathcal{V}_{\mathcal{G}^r} \equiv \mathcal{V}_\mathcal{K}$ and $\mathcal{E}_{\mathcal{G}^r} \equiv \mathcal{E}_\mathcal{K} \cup \{(u,v)\}$). Figure 3 illustrates the transformation. We then have*

$$V_{S,T}(\mathcal{G}) = V_{S,T}(\mathcal{G}^r).$$

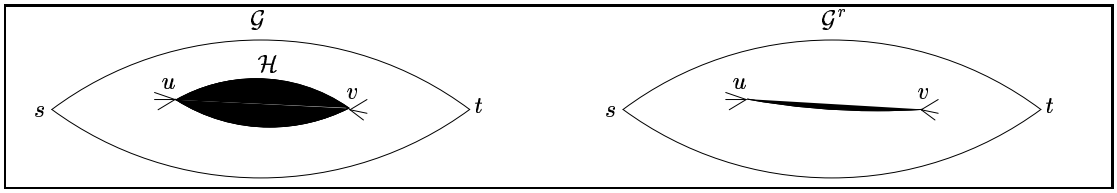

Figure 3: $\mathcal{G}$ and its reduced graph $\mathcal{G}^r$

The reader can see [1] for the proof of this theorem (in [1], the theorem is proven for (s,t)-mincuts, but the arguments are the same here). This result has a main consequence. It shows that the (S,T)-mincuts can be counted by a hierarchical decomposition of networks. That is, it makes possible to use a decomposition into triconnected components, leading to count (S,T)-mincuts in smaller networks. This decomposition can be used as a preprocessing on a large network built with several small networks. Then, each triconnected component is replaced by an edge, valued by its vector of mincuts.

Let us illustrate this property on the small example shown in Figure 4, which has several triconnected components. On this example, we want to evaluate the number of mincuts between the two white points. To do this, we show in the center of Figure 5 the reduced graph, where the three edges denoted by V_1, V_2 and V_3, are the vector of mincuts associated

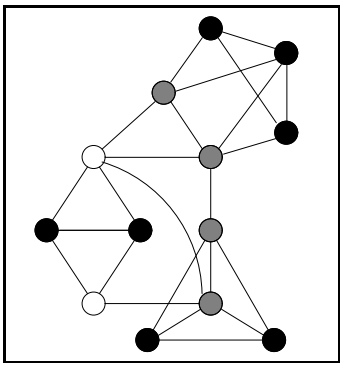

Figure 4: A biconnected graph with separation pairs, taken from [10]

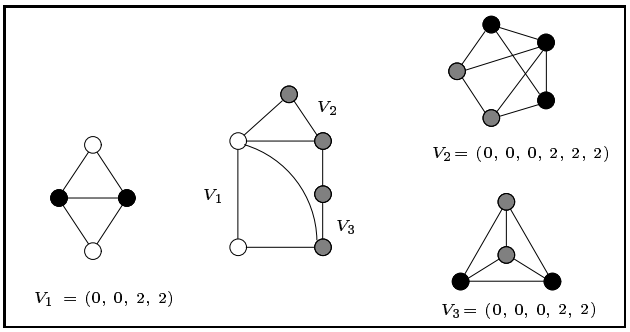

Figure 5: The vector of mincuts of three triconnected components of the graph of Figure 4 and the reduced associated graph

with the three triconnected components also shown in the same Figure. Then, by replacing two edges in series by one edge, and by repeating this operation two times, we obtain the network shown in Figure 6. After five last simplifications (parallel, series, parallel, series,

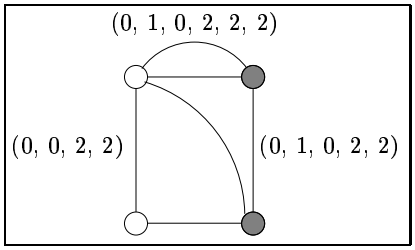

Figure 6: The graph obtained after two series reductions

parallel), we arrive at a single edge between the two marked nodes, weighted by the vector of (s, t)-mincuts of the original graph. The obtained vector is

$$(0, 0, 0, 2, 4, 4, 6, 12, 12, 8, 4, 0, \ldots).$$

4 The factoring algorithm

Below we give and prove the factoring theorem for the number of (S,T)-mincuts. This theorem gives a formula which is the basis of our factoring algorithm. This formula is similar to the relation of Moskowitz [9].

Theorem 4.1

Let $\mathcal{G} \equiv (\mathcal{V}, \mathcal{E})$ an undirected graph and $e \equiv (u,v)$ an edge. Then, we have

$$V_{S,T}(\mathcal{G}) = V_{S,T}(\mathcal{G} \bullet e) + w(e) * V_{S\cup\{u\},T\cup\{v\}}(\mathcal{G} - e) + w(e) * V_{S\cup\{v\},T\cup\{u\}}(\mathcal{G} - e).$$

Remark 4.2

- *If $u \in S$, then the formula becomes $V_{S,T}(\mathcal{G}) = V_{S_e,T}(\mathcal{G} \bullet e) + w(e) * V_{S,T\cup\{v\}}(\mathcal{G} - e)$*
- *If $u \in S$ and $v \in T$, then we have $V_{S,T}(\mathcal{G}) = w(e) * V_{S,T}(\mathcal{G} - e)$ and we obtain MP_4*
- *If $u, v \in S$, then the formula becomes $V_{S,T}(\mathcal{G}) = V_{S_e,T}(\mathcal{G} \bullet e)$ and we obtain MP_5*

Proof.

In the proof, we need to discuss about cuts in $\mathcal{G}$, in $\mathcal{G} \bullet e$ and in $\mathcal{G} - e$. To ease the reading of the proof, we explicitly denote by $(X, \overline{X})_{\mathcal{G}}$ the cut defined by $(X, \overline{X})$ in the graph $\mathcal{G}$. We also denote by $\mathcal{E}_{\mathcal{V}',\mathcal{G}}$ the set of edges of $\mathcal{G} \equiv (\mathcal{V}, \mathcal{E})$ having their extremities in $\mathcal{V}' \subseteq \mathcal{V}$. Thus, $(X, \mathcal{E}_{X,\mathcal{G}})$ is the graph induced by $X \subseteq \mathcal{V}$ in $\mathcal{G}$.
Let us denote, for fixed S, T, u, v,

$$\begin{aligned} MC_{\mathcal{G}}(u) \equiv \{\Gamma \equiv (X, \overline{X}) \mid\ & \Gamma \text{ is an } (S,T)\text{-mincut in } \mathcal{G} \\ & \text{with } u \in X,\ S \subseteq X,\ v \in \overline{X} \text{ and } T \subseteq \overline{X}\}, \\ MC_{\mathcal{G}}(v) \equiv \{\Gamma \equiv (X, \overline{X}) \mid\ & \Gamma \text{ is an } (S,T)\text{-mincut in } \mathcal{G} \\ & \text{with } v \in X,\ S \subseteq X,\ u \in \overline{X} \text{ and } T \subseteq \overline{X}\}, \end{aligned}$$

$$\overline{MC}_{\mathcal{G}} \equiv \{\Gamma \mid \Gamma \text{ is an } (S,T)\text{-mincut in } \mathcal{G} \text{ not separating } u \text{ and } v\},$$

$$\overline{MC}_{\mathcal{G}\bullet e} \equiv \{\Gamma \mid \Gamma \text{ is an } (S,T)\text{-mincut in } \mathcal{G} \bullet e\},$$

$$MC_{\mathcal{G}-e}(u) \equiv \{\Gamma \mid \Gamma \text{ is an } S',T'\text{-mincut in } \mathcal{G} - e \text{ with } S' \equiv S \cup \{u\} \text{ and } T' \equiv T \cup \{v\}\},$$

$$MC_{\mathcal{G}-e}(v) \equiv \{\Gamma \mid \Gamma \text{ is an } S',T'\text{-mincut in } \mathcal{G} - e \text{ with } S' \equiv S \cup \{v\} \text{ and } T' \equiv T \cup \{u\}\}.$$

Any (S,T)-mincut in $\mathcal{G}$ is in one (and only one) of the three subsets $MC_{\mathcal{G}}(u)$, $MC_{\mathcal{G}}(v)$ or $\overline{MC}_{\mathcal{G}}$.

(i) Let $\Gamma \equiv (X,\overline{X})_{\mathcal{G}} \in \overline{MC}_{\mathcal{G}}$, with, say, $u,v \in X$. Let us denote by X' the set of nodes X where u and v are replaced by w, we have that $\overline{X}' = \overline{X}$ and $(X,\overline{X})_{\mathcal{G}} = (X',\overline{X}')_{\mathcal{G}\bullet e}$, which is an (S,T)-cut in $\mathcal{G}$ and in $\mathcal{G}\bullet e$. The minimality of $(X,\overline{X})_{\mathcal{G}}$ in $\mathcal{G}$ implies that $(X,\mathcal{E}_{X,\mathcal{G}})$ and $(\overline{X},\mathcal{E}_{\overline{X},\mathcal{G}})$ are both connected, then $(X',\mathcal{E}_{X',\mathcal{G}\bullet e})$ and $(\overline{X}',\mathcal{E}_{\overline{X}',\mathcal{G}\bullet e})$ are connected as well. So, $(X',\overline{X}')_{\mathcal{G}\bullet e}$ is minimal. Conversely, if $(X',\overline{X}')_{\mathcal{G}\bullet e}$ is minimal, then $(X,\overline{X})_{\mathcal{G}}$ is minimal. We can deduce that there is a one-to-one correspondence between $\overline{MC}_{\mathcal{G}}$ and $\overline{MC}_{\mathcal{G}\bullet e}$:

$$\Gamma \in \overline{MC}_{\mathcal{G}\bullet e} \Longleftrightarrow \Gamma \in \overline{MC}_{\mathcal{G}}. \tag{1}$$

(ii) Observe that there is a one-to-one correspondence between $MC_{\mathcal{G}}(u)$ and $MC_{\mathcal{G}-e}(u)$, and between $MC_{\mathcal{G}}(v)$ and $MC_{\mathcal{G}-e}(v)$. Indeed,

$$\Gamma \in MC_{\mathcal{G}-e}(u) \Longleftrightarrow \Gamma + e \in MC_{\mathcal{G}}(u). \tag{2}$$

$$\Gamma \in MC_{\mathcal{G}-e}(v) \Longleftrightarrow \Gamma + e \in MC_{\mathcal{G}}(v). \tag{3}$$

From these three equivalence and the results on the triconnected components, we can deduce

$$V_{S,T}(\mathcal{G}) = V_{S,T}(\mathcal{G}\bullet e) + w(e) * V_{S\cup\{u\},T\cup\{v\}}(\mathcal{G}-e) + w(e) * V_{S\cup\{v\},T\cup\{u\}}(\mathcal{G}-e),$$

which ends the proof. ■

Remark 4.3 *The proof is based on the fact that the (S,T)-mincuts are unchanged in the subgraphs. Then, this formula could be used in order to generate all the (S,T)-mincuts.*

Based on this theorem, we can propose an algorithm which selects an edge e of $\mathcal{G}$ and uses a pivotal decomposition on this edge. Then, we apply the factoring theorem in order to count the (S,T)-mincuts in each subgraph deduced from $\mathcal{G}$. We present here a procedure which depends on the strategy used to choose e. An edge-selection strategy is a set of rules that indicates which edge e is selected. In the next section, we will give an example of such a strategy.

Input: an undirected graph $\mathcal{G}$ and two vertices s and t.

Output: the vector of (s,t)-mincuts of $\mathcal{G}$; that is the number of (s,t)-mincuts of each size.

Procedure $COUNT(\mathcal{G}, S, T)$

$\Omega \leftarrow 1$

If ALLCUTS, then return $V_{S,T} \leftarrow (1, 0, 0, \dots)$

If NOCUTS, then return $V_{S,T} \leftarrow (0, 0, 0, \dots)$

Until no *MP* can be made do

Perform a reduction MP_i on $\mathcal{G}$ to obtain $\mathcal{G}'$, S' and T'

Evaluate α such that $V_{S,T}(\mathcal{G}) \equiv \alpha * V_{S',T'}(\mathcal{G}')$

$\Omega \leftarrow \alpha * \Omega$

$\mathcal{G} \leftarrow \mathcal{G}'$, $S \leftarrow S'$ and $T \leftarrow T'$

If $\mathcal{G}$ is an edge e then return $\Omega * V_{S,T}(e)$

Select an edge $e \equiv (u, v)$ using some strategy $STRAT$

$V_1 \leftarrow COUNT(\mathcal{G} \bullet e, S, T)$

$V_2 \leftarrow COUNT(\mathcal{G} - e, S \cup \{u\}, T \cup \{v\})$

$V_3 \leftarrow COUNT(\mathcal{G} - e, S \cup \{v\}, T \cup \{u\})$

$V_{S,T}(\mathcal{G}) \leftarrow \Omega * (V_1 + w(e) * V_2 + w(e) * V_3)$

Return $V_{S,T}(\mathcal{G})$

We have ALLCUTS $\equiv 1$, if $\mathcal{G}$ is disconnected with S included in one connected component and T included in an other one.

We have NOCUTS $\equiv 1$, if $S \cap T \neq \emptyset$, if there is no S-tree and T-tree having no common point or if there is no S-tree or no T-tree.

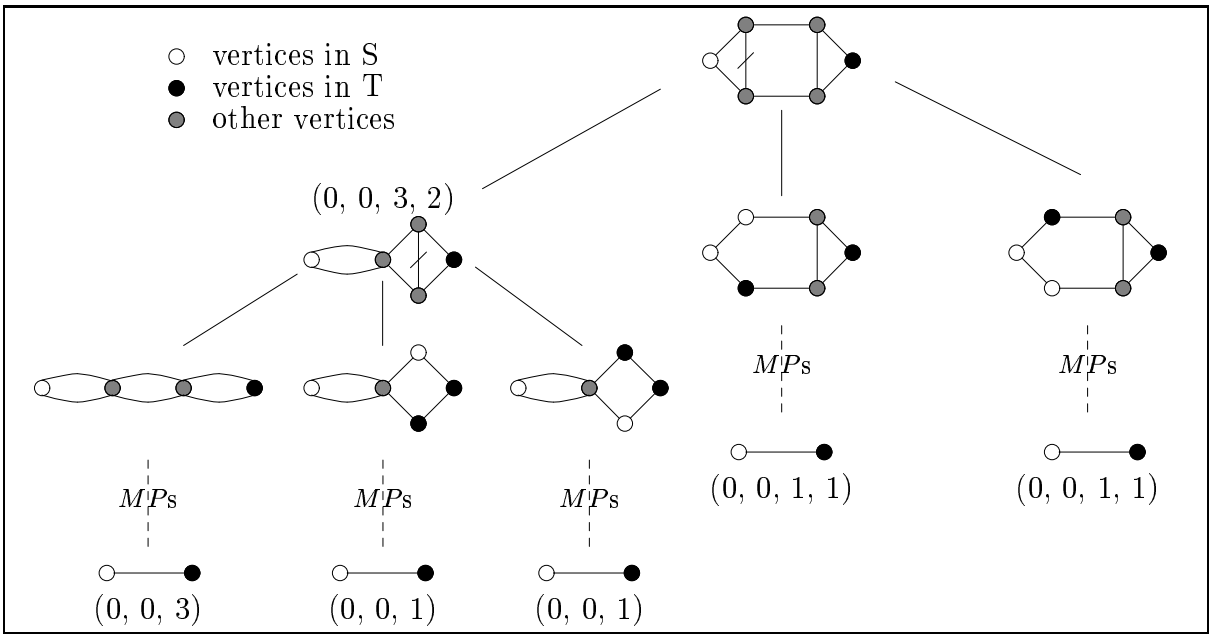

Figure 7: An example of factorization, with $V_{s,t}(\mathcal{G}) = (0, 0, 3, 4, 2)$

5 Comparison with a previous algorithm

Generating and counting all (s,t)-mincuts in undirected graphs has been studied in many papers. According to their complexity results, one of the most efficient known approach has been developed by Provan and Shier in [2] (similar to the one developed by Tsukiyama and *al.* in [3]). The algorithm is based on a pivotal decomposition on a node v and its pivot set $I(S,v)$. A pivot node relative to the set $S \subseteq \mathcal{V}$ is any node $v \notin S$ which is a successor of a node in S. The associated pivot set is $I(S,v) \equiv \{u \in \overline{S} \mid \text{every } (u,t)\text{-path in } \langle \overline{S} \rangle \text{ contains } v\}$. We give here the algorithm and show that it can be viewed as an edge-factoring algorithm with no series-parallel reductions. The main procedure, defined for two subsets of nodes S and T, is the following:

Procedure $LIST(S,T)$

PIVOT$(S,T,v,I(S,v))$;

if $I(S,v)=\emptyset$ **then** output the cut $(S,\overline{S})$

else

LIST$(S,T\cup\{v\})$

LIST$(S\cup I(S,v),T)$

The procedure PIVOT, relative to sets S and T, is based on the biconnected components. It can be implemented as follows:

1. Construct the graph $\tau\equiv(\mathcal{V}_\tau,\mathcal{E}_\tau)$ whose vertex set $\mathcal{V}_\tau$ consists of $\overline{S}$ together with a vertex b_i for each biconnected component B_i of $\langle\overline{S}\rangle$. The edge set $\mathcal{E}_\tau$ contains all pairs (v,b_i) with $v\in B_i$. It is easy to see that τ is a tree.

2. In the tree τ, identify the set M of those vertices v, which are a successor of S, that are maximally distant from t: that is, for which the set $\mathcal{V}(v)$ of vertices in $\overline{S}$ separated from t by v contains no other successor of S. (Note that if there exists a pivot element v satisfying $I(S,v)\subseteq\overline{T}$, then one which lies in M must exist.)

3. For each $v\in M$ check whether $\mathcal{V}(v)\subseteq\overline{T}$. If this hold for some v, then return v and $I(S,v):=\mathcal{V}(v)$; otherwise return $I(S,v):=\emptyset$.

LIST$(S,T\cup\{u\})$ and $COUNT(\mathcal{G},S,T\cup\{u\})$ give the same result. If there are parallel edges from s to u, we perform MP_1, until there is only one edge e, valued by $V_{s,u}(e)$. Then, by performing MP_4, with the edge $e\equiv(s,u)$, we obtain that $COUNT(\mathcal{G},S,T\cup\{u\})=V_{s,u}(e)*COUNT(\mathcal{G}-e,S\cup\{s\},T\cup\{u\})$. Then, LIST$(S,T\cup\{u\})$ gives the same result as $V_{s,u}(e)*V_2$

LIST$(S\cup I(S,u),T)$ and $COUNT(\mathcal{G},S\cup I(S,u),T)$ give the same result. Suppose that S contains only one element, s. Then, as s and u are adjacent, by performing MP_5, we obtain that $COUNT(\mathcal{G},\{s\}\cup I(s,u),T)=COUNT(\mathcal{G}\bullet e,\{s\}\cup I(s,u),T)$. From its construction, $I(s,u)$ is a biconnected component of $\mathcal{G}\bullet e$, with the cutvertex s, that does not contain any element of T. By performing MP_7, we obtain that $COUNT(\mathcal{G}\bullet e,\{s\}\cup I(s,u),T)=$

$COUNT(\mathcal{G} \bullet e, s, T)$. Moreover, we obtain that S still contains only one element and since we do not perform any MP that modifies S, we have always that $S = s$.

If $I(S, v) = \emptyset$, we have several possibilities:

- Either all the successors of S, v_j, are in T. We can have multiple edges from s to v_j. We denote them e_j^i. By performing MP_4 for each edge adjacent to s, we obtain $V_{S,T}(\mathcal{G}) = V_{s,v_j}(e_j^i) * V_{S,T}(\mathcal{G} - e_j^i)$. From the choice of each pivot vertex v, $\langle S \rangle$ and $\langle \overline{S} \rangle$ are always connected. When there is no edge adjacent to s, we obtain $V_{S,T}(\mathcal{G}') = (1, 0, \ldots, 0)$. Then, we have $V_{S,T}(\mathcal{G}) \equiv \mathop{\ast}_i V_{s,v_j}(e_j^i)$, which is the number of edges of $(S, \overline{S})$.

- Either there is a vertex $v \notin T$, but $I(S, v) \cap T \neq \emptyset$. Then, by performing MP_6, we obtain $v \in T$ and we are in the previous case where all the successors of S are in T.

Let us consider the factoring algorithm and define the following strategy $STRAT$ to select the pivotal edge by:

- select a pivotal vertex u using the procedure of Provan and Shier
- If there are parallel edges from s to u, perform MP_1
- then, select the pivotal edge (s, u).

By using this strategy $STRAT$ and the mincut-preserving reductions MP_4, MP_5, MP_6 and MP_7, we obtain the procedure of Provan and Shier, whose complexity is in $O(m||V_{s,t}(\mathcal{G})||)$ where m denotes the number of edges and $||V_{s,t}(\mathcal{G})||$, the number of (s,t)-mincuts. So, we have the following property.

Property 5.1 *The paradigm of Provan and Shier belongs to the family of edge-factoring algorithms.*

Let us point out again the interest in performing series-parallel simplifications. For instance, in the case of the well known example shown in Figure 8, our algorithm runs ≈ 50 times faster by performing the series-parallel reductions. Let us consider the binary decomposition tree associated with the strategy of Provan and Shier, defined as follows: we

associate with the root of the tree, the graph $\mathcal{G}$, and each node $\mathcal{G}_0$ is associated with a call to the procedure $LIST$. Each leaf of the tree corresponds to an (s,t)-mincut in $\mathcal{G}$. So, the number of nodes is equal to $2 \times NC - 1$ where NC denotes the number of (s,t)-mincuts in $\mathcal{G}$. When we apply the factoring algorithm with no MP reductions to the topology of Arpanet given in Figure 8, we obtain a tree with 1171 nodes. We obtain only 21 nodes by performing series-parallel reductions.

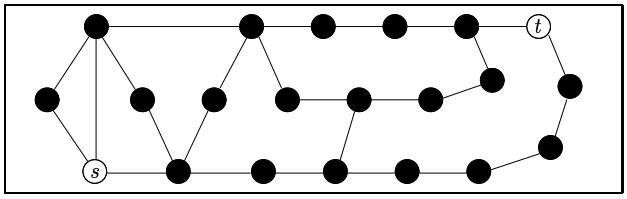

Figure 8: A version of the Arpanet topology

The problem of counting all the (s,t)-mincuts is an NP-hard problem [7]. However, in most of practical cases, this can be done very quickly. We have count the number of mincuts between the two black squares in the TRANSPAC French packet switching network presented in Figure 9, which has 28 nodes and 95 edges. By using the generation algorithm of Provan and Shier, we need about 9 hours to count all the (s,t)-mincuts (5,483,994 mincuts), whereas our algorithm (with series parallel reductions) needed only 12 minutes to count all the (s,t)-mincuts.

6 Conclusions

We have studied the problem of computing the number of (s,t)-mincuts of size i for all i, in a network. To this purpose, we propose an algorithm based on a factoring formula which transforms the problem on a graph into three problems on smaller graphs. The advantage of this method is the possibility to use mincut-preserving reductions, with computational cost polynomial in the size of the network, in order to diminish the size of the problems. We show that the most efficient known method to generate all the (s,t)-mincuts can be viewed as an edge-factoring algorithm with no mincut-preserving reductions. This work can be extended to the problem of generating (that is not only counting) all the (s,t)-mincuts. The main

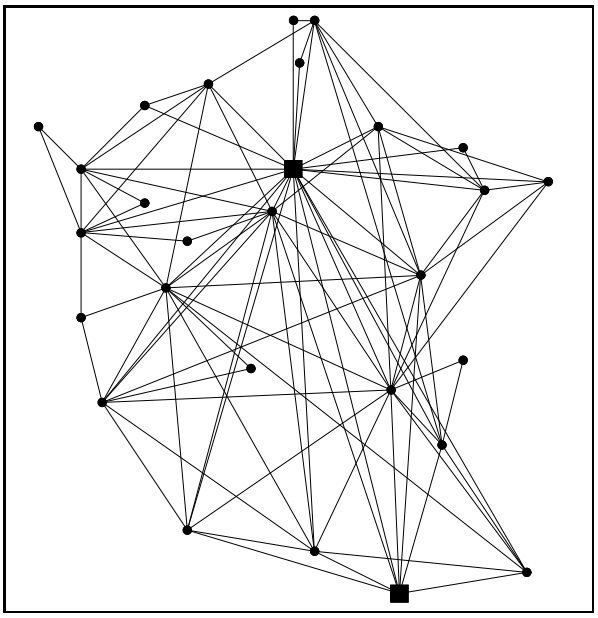

Figure 9: A version of the TRANSPAC network packet switching network

problem to be solved seems to find an efficient data storage scheme. Current research work is being done in this direction.

References

[1] S. Bulteau and G. Rubino. A new approach to vulnerability evaluation of communication networks. In *Proceedings of the International Teletraffic Congress (ITC) 15*, Washington, D.C., U.S.A., June 1997. An extended version can be found at http://www.irisa.fr/EXTERNE/bibli/pi/1089/1089.html.

[2] J.S. Provan and D.R. Shier. A paradigm for listing (s,t)-cuts in graphs. *Algorithmica*, 15:351–372, 1996.

[3] S. Tsukiyama and Z. Shirakawa and H. Ozaki and H. Ariyoshi. An algorithm to enumerate all cutsets of a graph in linear time per cutset. *J.ACM*, 27:619–631, 1980.

[4] C.S. Sung and B.Y. Yoo Simple enumeration of minimal cutsets separating 2 vertices in a class of undirected planar graphs. *IEEE Transactions on Reliability*, 41:62–71, 1992.

[5] A. Ramanathan and C.J. Colbourn. Counting almost minimum cutsets with reliability applications. *Mathematical Programming*, 39:253–261, 1987.

[6] K. Wood. A factoring algorithm using polygon-to-chain reductions for computing k-terminal network reliability. *Networks*, 15:173–190, 1985.

[7] J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal of Computing*, 12:777–788, 1983.

[8] A. Satyanarayana and K. Wood. A linear-time algorithm for computing k-terminal in series-parallel networks. *SIAM Journal of Computing*, 14:818–832, 1985.

[9] F. Moskowitz. The analysis of redundancy networks. *AIEE Trans. (Commun. Electron.)*, 39:627–632, 1958.

[10] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal of Computing*, 2:135–158, 1973.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399